Relazione Progetto
Corso Sistemi Distribuiti

Antonio Cardace, Michele Cucchi, Federico Fossemò
Corso Informatica Magistrale
Università di Bologna

Aprile 2016

1 Abstract

Il progetto implementa la realizzazione in modalità distribuita multinodo, del gioco di carte da tavolo UNO.
La relazione descrive in modo particolare i problemi affrontati, le soluzioni proposte, quelle scartate e l'implementazione definitiva.

2 Introduzione

3 Aspetti progettuali

3.1 Introduzione

Il gioco sviluppato rispetta esattamente le regole del gioco da tavolo descritte successivamente, si svolge a turni e per scelta progettuale prevede un minimo di 4 ed un massimo di 8 giocatori, ognuno dei quali si trova in un singolo host separato.
La presentazione ed il tavolo di gioco sono costruiti graficamente rispecchiando fedelmente forma e simbologia delle carte. La costruzione grafica cerca di approssimare un tavolo da gioco reale.
Il sistema di gioco prevede due fasi principali di svolgimento in cui il paradigma di comunicazione tra gli host cambia completamente.
Nella fase di inizializzazione il gioco parte con i nodi dei giocatori attivi in modalità **client-server**, per sfruttare le funzionalità di registrazione dei giocatori fornite da un nodo server, che genera e distribuisce gli identificativi numerici univoci degli host giocatori.
Nella fase di gioco principale il paradigma di comunicazione tra i nodi cambia passando ad una modalità completamente **peer2peer**, in cui ogni host mantiene la visione del gioco per il proprio giocatore e si sincronizza con gli altri per mantenere lo stato consistente, anche nel caso di crash o down di nodi.

3.2 Descrizione del gioco

Un giocatore può vincere una partita di UNO quando rimane senza alcuna carta in mano, quindi l'obiettivo del gioco è perdere più carte possibile nel tempo più breve possibile.
All'inizio del gioco il mazziere assegna 7 carte a caso a ciascun giocatore, lasciando le restanti carte in un mazzo sul tavolo a dorso coperto.
La prima carta della pila viene lasciata visibile, perchè sarà la carta iniziale del gioco, formando la prima della pila scarti. I giocatori procedono a turni, in senso orario, a partire da quello alla sinistra del mazziere, lasciando sul tavolo una carta delle proprie sette, che abbia stesso colore o stesso numero di quella lasciata scoperta. c'è la possibilità di utilizzare una carta speciale, ma se sono colorate devono essere compatibili con quella scoperta. Non è possibile scartare più di una carta per turno e nel caso un giocatore non abbia carte giocabili, deve prenderne una dal mazzo, se è giocabile la dovrà spendere immediatamente altrimenti passerà il turno.
Esistono comunque carte speciali che producono vantaggi o svantaggi per il giocatore.

- **Carta Divieto** provoca la perdita del turno di gioco al giocatore successivo, nel caso di due soli giocatori il giocatore che trova la carta può rigiocare immediatamente
- **Carta Inversione** provoca l'inversione del senso dei turni di gioco, fino alla prossima uscita della carta
- **Carta +2** impone al giocatore del turno successivo di prendere 2 carte
- **Carta +4** impone al giocatore successivo di prendere 4 carte, chi gioca la carta può scegliere il prossimo colore
- **Carta Jolly** giocabile in ogni momento, consente a chi l'ha giocata di decidere il colore giocabile dal giocatore successivo

3.3 Implementazione distribuita

L'implementazione distribuita del gioco descritto impone l'analisi e la gestione dei problemi che ne derivano, in particolare emergono alcuni problemi specifici:

3.3.1 Registrazione ed identificazione giocatori

Gli host dei giocatori in fase di gioco avviato comunicano tra loro direttamente secondo il paradigma **peer2peer**, ma prima di avviare il gioco è necessaria una sincronizzazione dei giocatori presso un punto comune, anche per ragioni di identificazione reciproca. La sincronizzazione iniziale viene realizzata tramite un'implementazione temporanea, solo per l'avvio, di un sistema **client-server**. Il server è l'host comune, presso il quale i nuovi giocatori devono registrarsi, per *iscriversi* al gioco. L'operazione di registrazione restituisce anche un identificativo numerico univoco, che consente di riconoscere inequivocabilmente i giocatori.

Al termine delle operazioni di registrazione, una volta soddisfatta la condizione del numero minimo di giocatori, il gioco verrà avviato e di conseguenza gli host si scambieranno informazioni direttamente tra loro, rendendo superflua la presenza del server.

3.3.2 Gestione dei turni di gioco

L'andamento del gioco procede a turni si rende quindi necessario un paradigma di sincronizzazione tra i vari nodi giocatori, per stabilire inequivocabilmente chi è abilitato al gioco in ogni momento.
La sincronizzazione avviene grazie ad un *token* che viene fatto passare tra gli host in sequenza, realizzando virtualmente *l'anello* dei partecipanti al gioco. La direzione di partenza è stabilita secondo le regole del gioco.
Un giocatore assume il turno quando entra in possesso del token dal nodo che ha appena giocato, al termine del turno spedisce il token al giocatore successivo.
La gestione dei turni risulta molto affidabile ed *autogestita* dagli stessi nodi senza bisogno di entità esterne alla rete peer2peer.

3.3.3 Gestione crash nodi e politica di fault-tolerance

L'implementazione distribuita del gioco realizzata tramite nodi che ne eseguono una copia e comunicano tramite una rete utilizzando una sorta di protocollo peer2peer, è vulnerabile, per la natura stessa del sistema, in caso di perdita della comunicazione tra i nodi o al crash di uno o più di essi.
La perdita di uno più nodi, infatti nel caso specifico del gioco UNO, potrebbe provocare l'impossibilità di proseguire il gioco, ad esempio come nel caso di crash del nodo successivo al termine di un turno oppure dello stesso host attualmente in turno di gioco.
La realizzazione di contromisure in grado di rendere trasparenti al resto dei nodi la perdita di uno o più host, risulta quindi obbligatoria.
Sono state analizzate due possibili soluzioni:

1. **Leader**: eletto tra i nodi, gestisce il fault tolerance
2. **Token di Fault-Tolerance**: gira tra i nodi per gestire in modo peer2peer, il crash di nodi.

4 Aspetti implementativi

5 Valutazione

6 Conclusioni